Machine Learning

Lesson 9: Recommender Systems

n.vimalkumar@hotmail.com

Concepts Covered

- Theory of Recommender Systems
- Collaborative Filtering
- User Based Nearest Neighbour
- Item Based Nearest Neighbour
- Cosine and Adjusted Cosine Similarity
- Association Rule Mining
- Apriori Algorithm

n.vimalkumar@hotmail.com

Learning Objectives

By the end of this lesson, you will be able to:

- Build recommender model using python
- Understand mechanism of association rule mining
- Demonstrate apriori algorithm

n.vimalkumar@hotmail.com

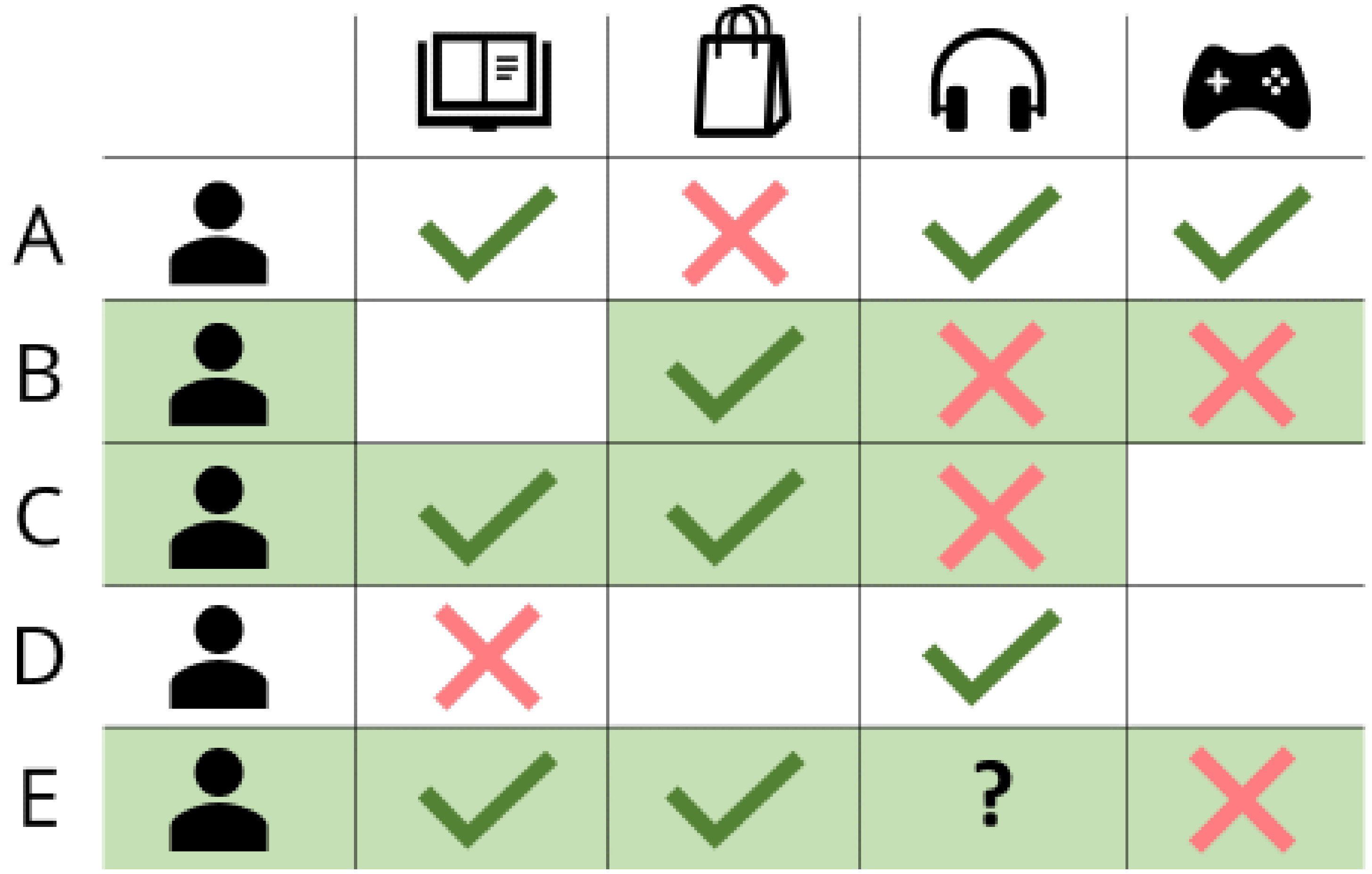
Recommender Systems

Topic 1: Introduction

n.vimalkumar@hotmail.com

Recommender Systems

Recommender system is an information filtering technique, which provides users with recommendations, which they might be interested in.



n.vimalkumar@hotmail.com

Recommender Systems: Solution

Recommender systems acts as a solution for your day to day choices

Which websites will you find interesting?

Which digital camera should you buy?

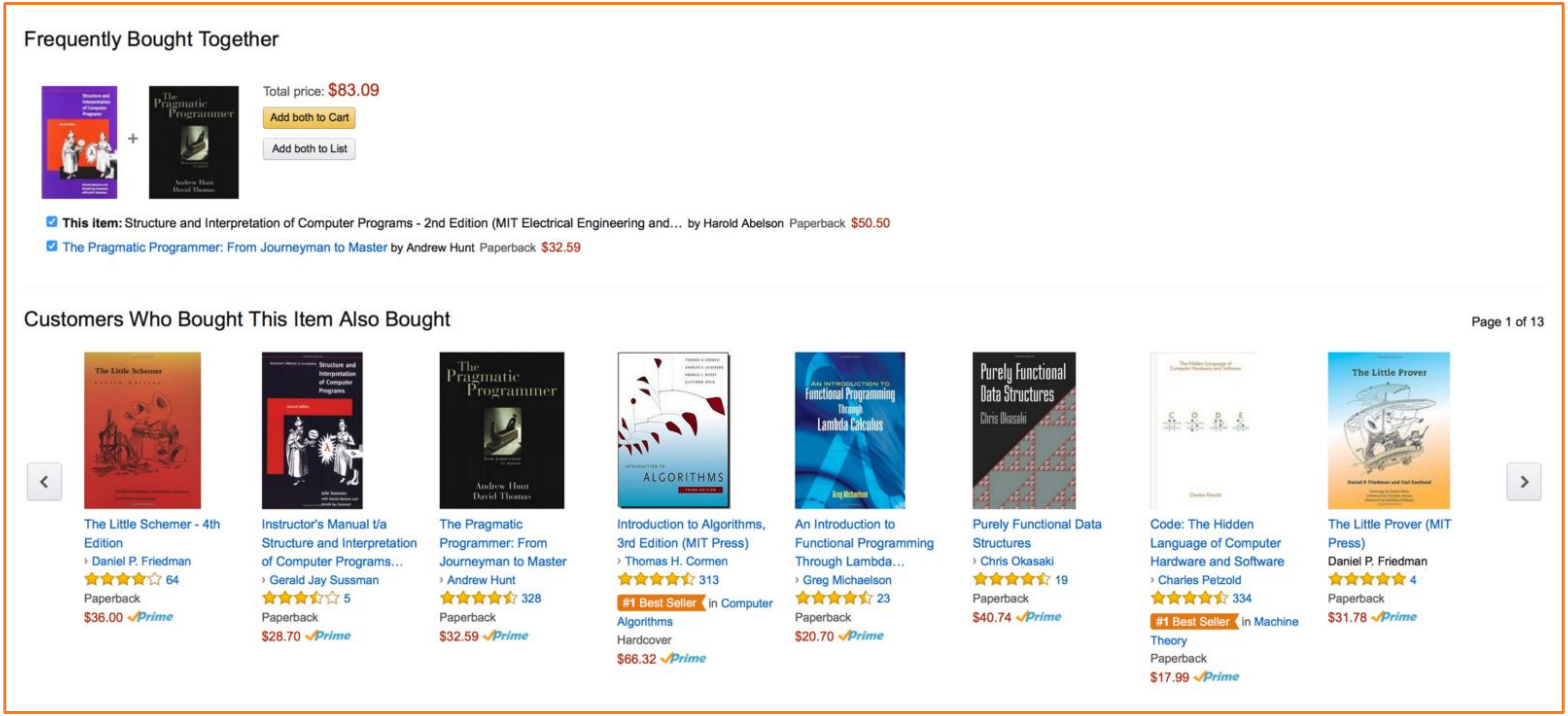
Which degree and university are best for your future?

Which book should you buy for your next vacation?

Which is the best investment for supporting the education of your children?

What is the best holiday for you and your family?

Recommender Systems: Example



Success of Recommendation Systems

Success and Purpose of Recommendation Systems

Prediction Perspective
- Predicts to what degree a user likes an item
- Most popular evaluation scenario in research

Interaction Perspective
- Gives users a good feeling
- Educates users about the product domain

Conversion Perspective
- Commercial Situations
- Increases hit and clickthrough, lookers to bookers rates

Different Aspects
- Depends on domain and purpose
- No holistic evaluation scenario exists

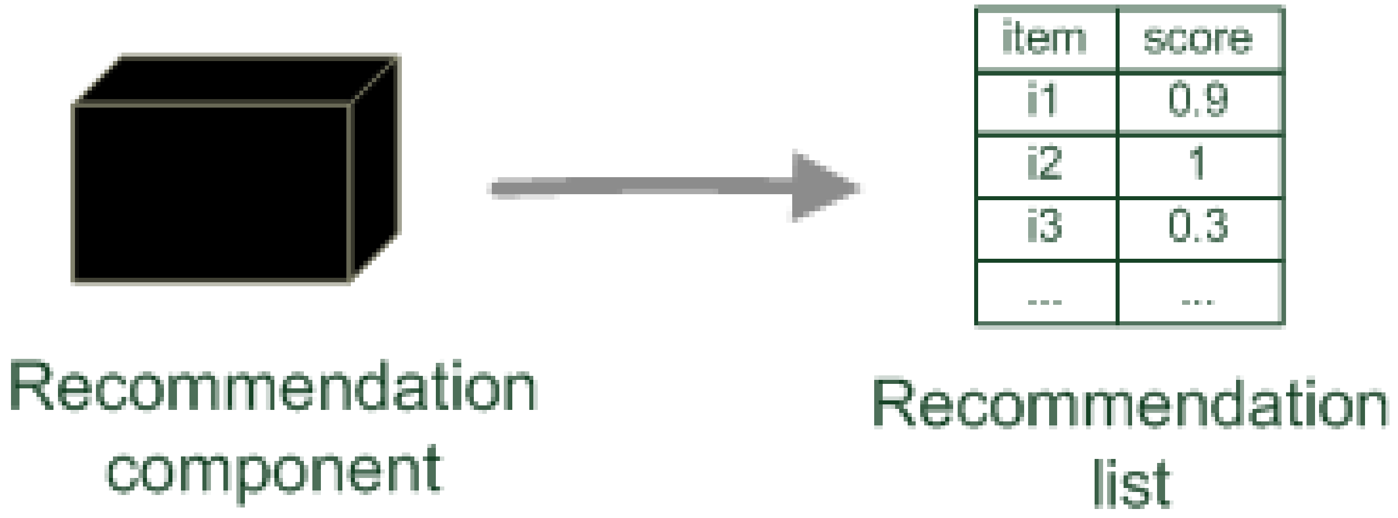
Retrieval Perspective
- Reduces search costs
- Provides correct proposals

Recommendation Perspective
- Identifies items from the long tail
- Users did not know about their existence

n.vimalkumar@hotmail.com

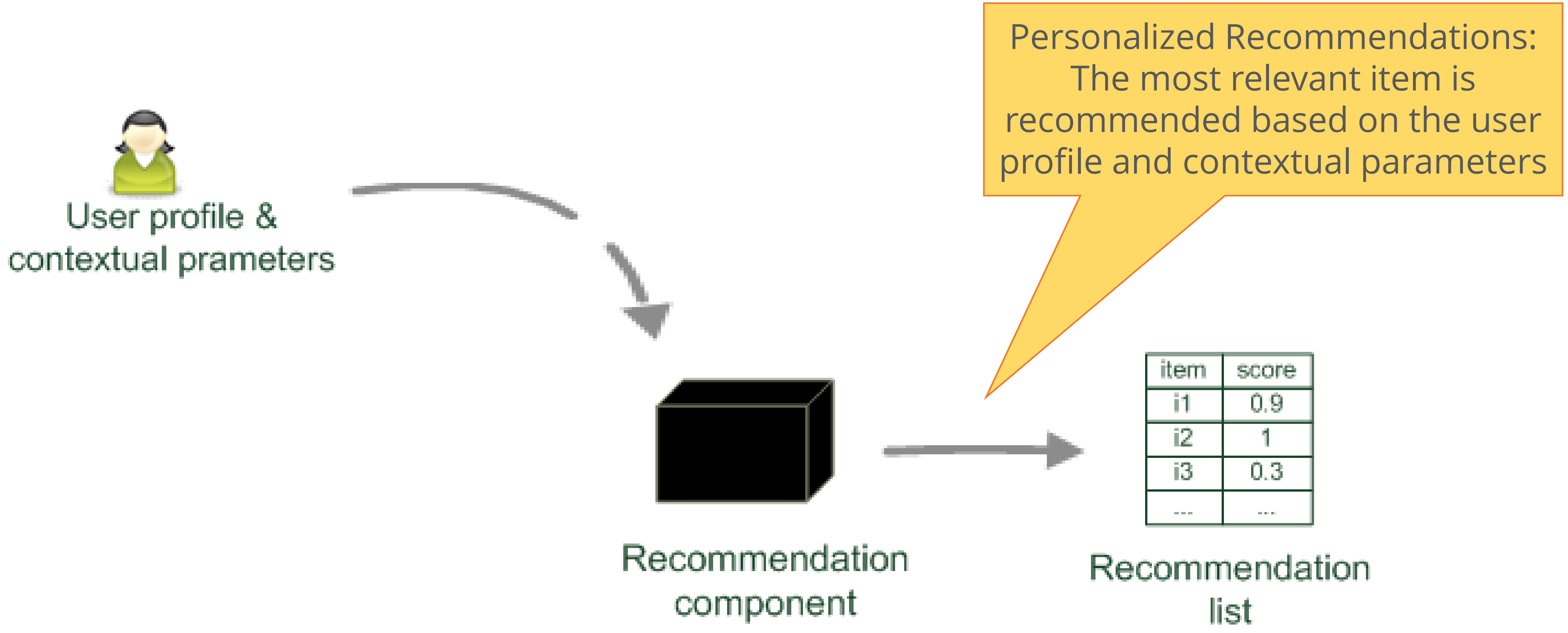
Paradigms of Recommendation Systems

Recommender systems reduce information overload by estimating relevance



n.vimalkumar@hotmail.com

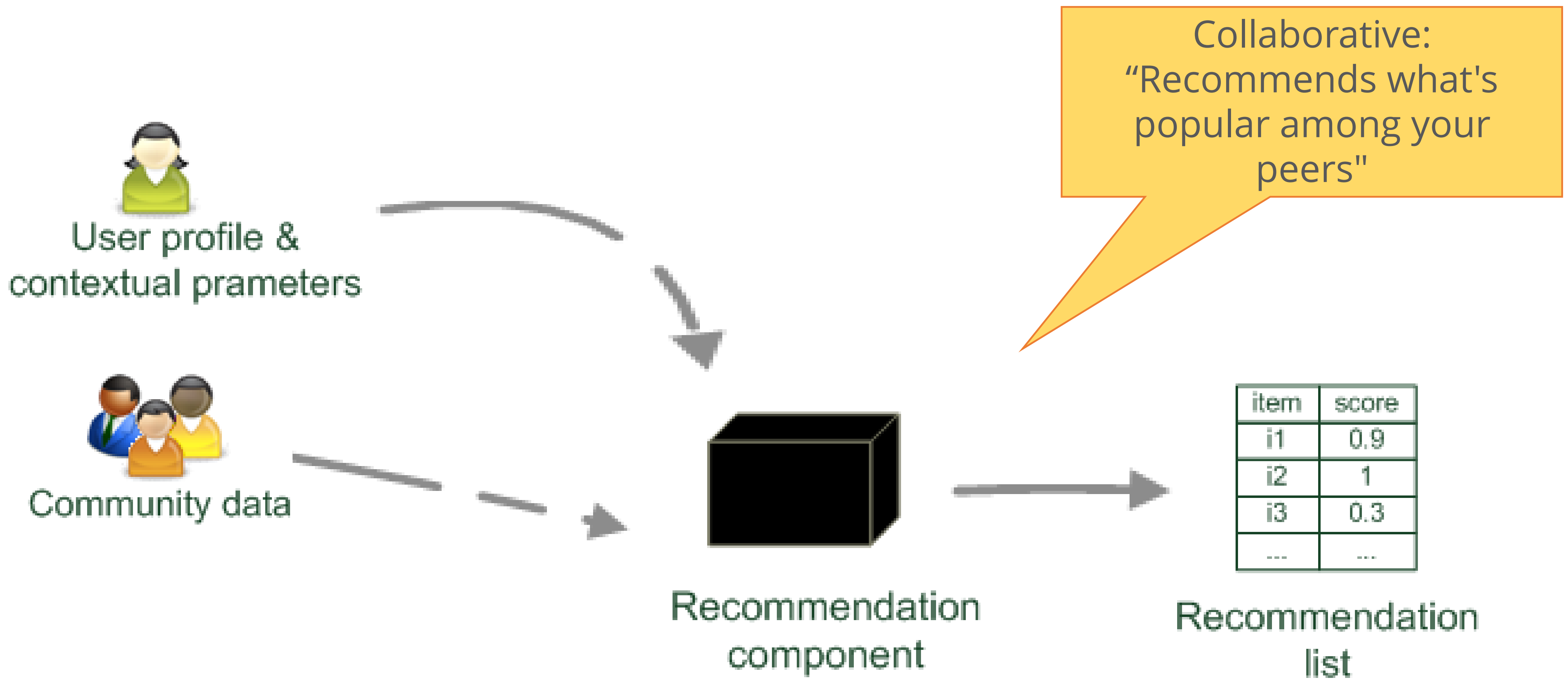
Paradigms of Recommendation Systems (Contd.)



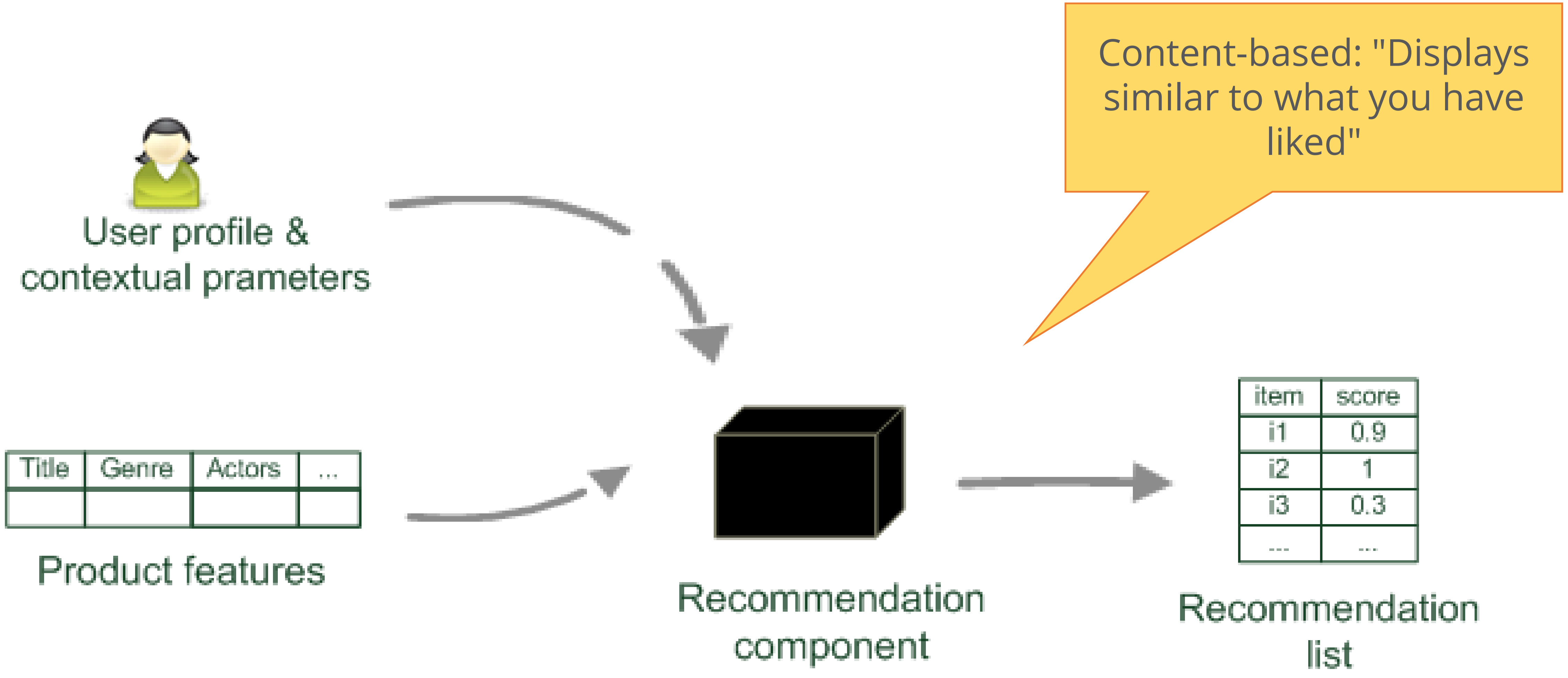
item	score
i1	0.9
i2	1
i3	0.3
...	...

n.vimalkumar@hotmail.com

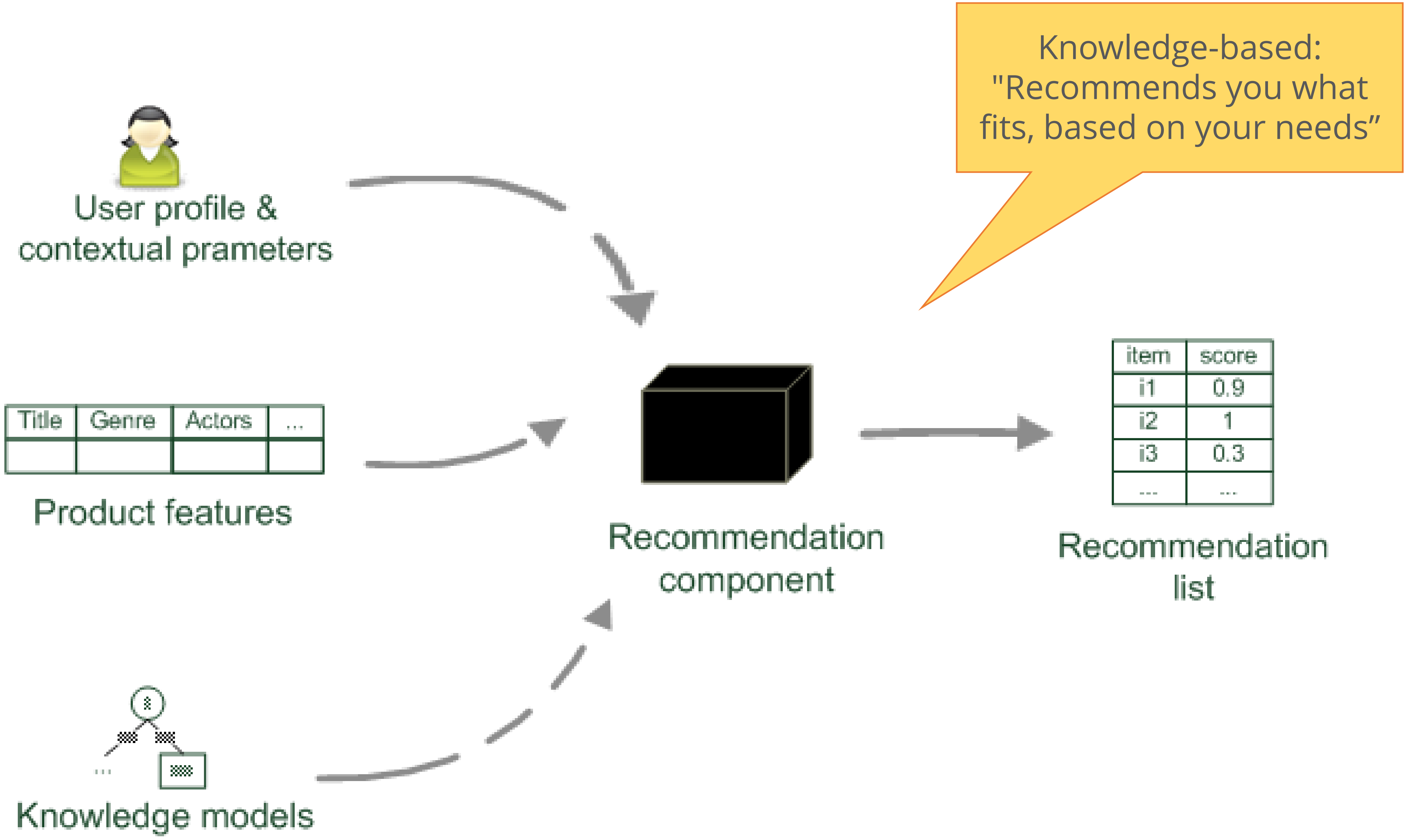
Paradigms of Recommendation Systems (Contd.)



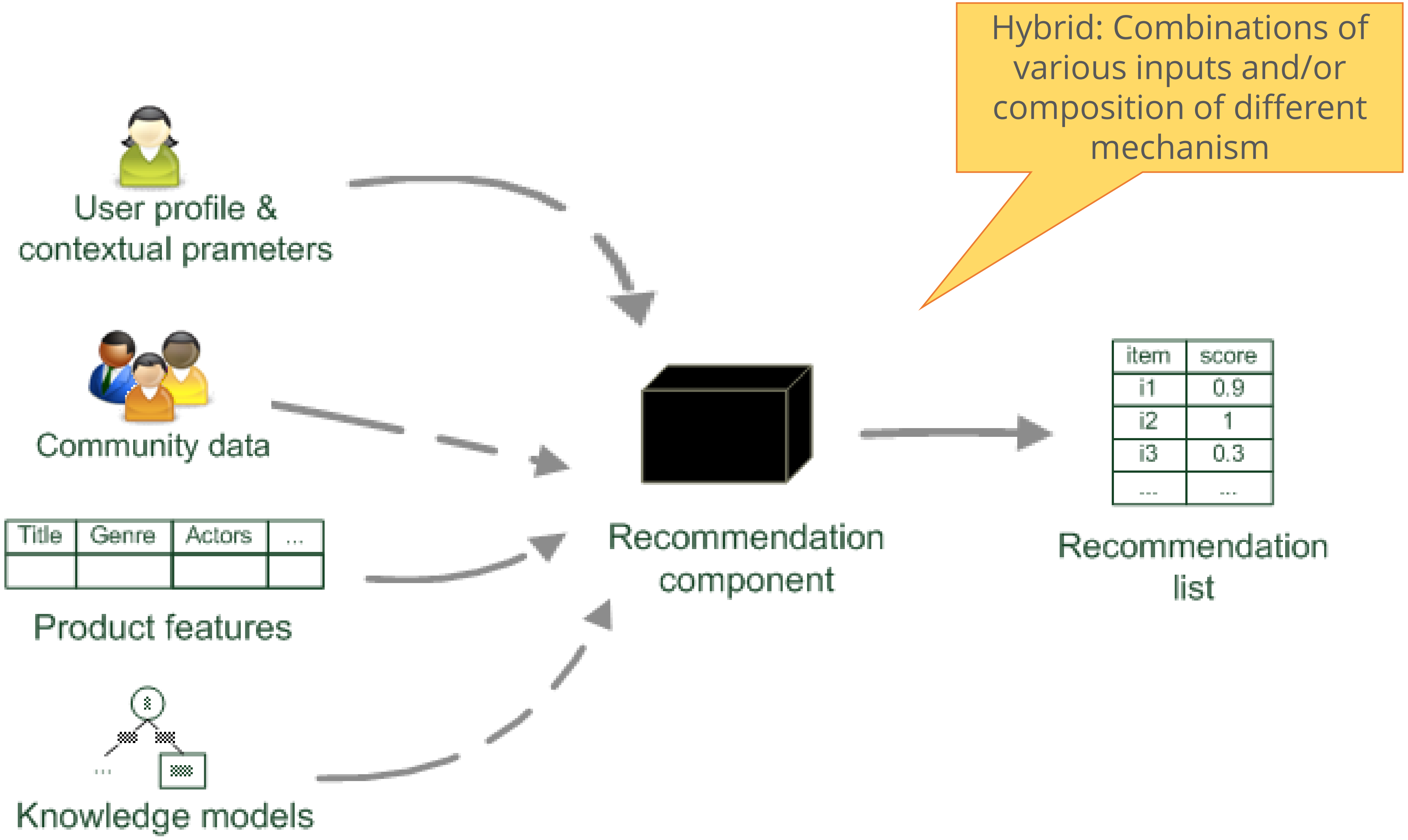
Paradigms of Recommendation Systems (Contd.)



Paradigms of Recommendation Systems (Contd.)



Paradigms of Recommendation Systems (Contd.)



Recommender Systems

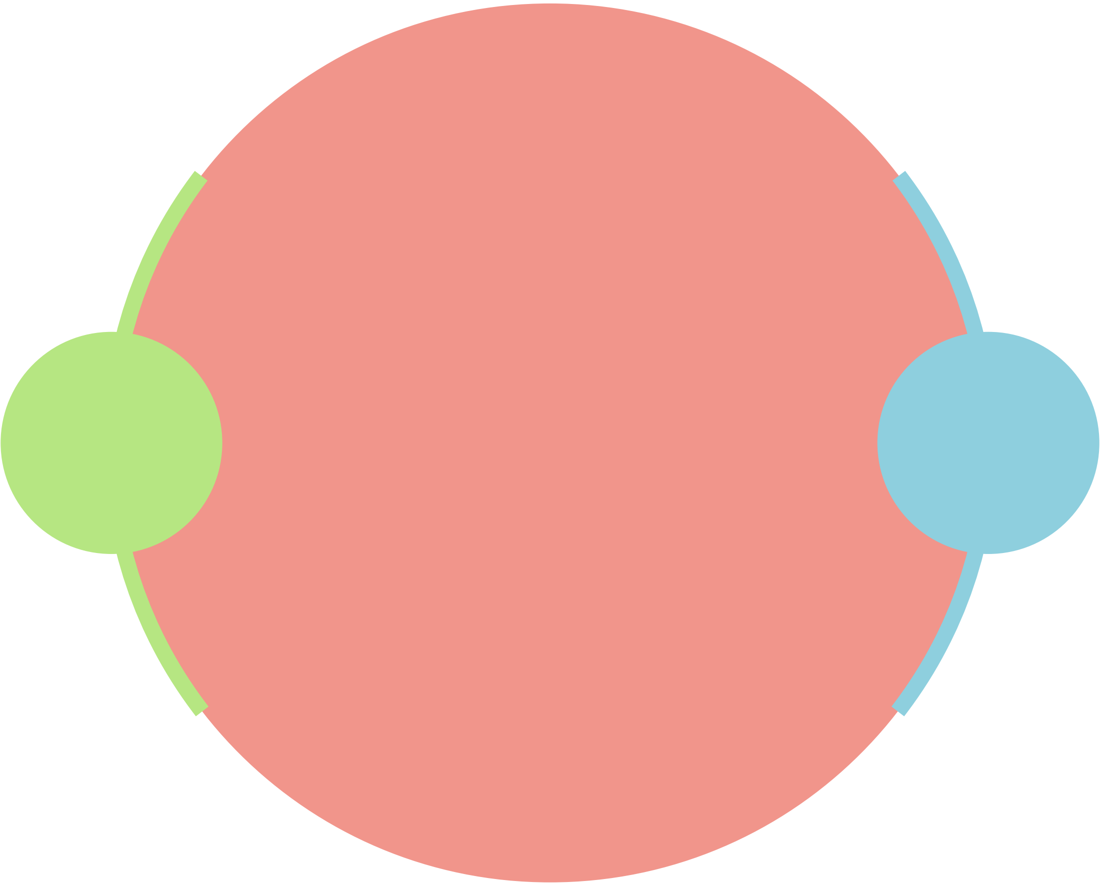
Topic 2: Collaborative Filtering

n.vimalkumar@hotmail.com

Collaborative Filtering

It matches people with similar interests as a basis for recommendation.

Item-Based Nearest Neighbor



User-Based Nearest Neighbor

User-Based Nearest Neighbor

Consider a database of ratings of the current user, Alice and other users given:

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Problem

Determine whether Alice will like the Item 5 that is not seen or rated

Measuring User Similarity: Pearson Correlation

A measure of how strong a relationship is between two variables

$$\boldsymbol{sim(a,b)} = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}$$

Where,

a, b : users

$r_{a,p}$: rating of user a for item p

P : set of items, rated both by a and b

Possible similarity values between -1 and 1

n.vimalkumar@hotmail.com

Measuring User Similarity: Pearson Correlation

A common prediction function

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

sim = 0,85

sim = 0,00

sim = 0,70

sim = -0,79

Making Predictions

$$\mathbf{pred}(\mathbf{a},\mathbf{p}) = \bar{r}_a + \frac{\sum_{\mathbf{b} \in N} \mathbf{sim}(\mathbf{a},\mathbf{b}) * (\mathbf{r}_{\mathbf{b},\mathbf{p}} - \bar{\mathbf{r}}_{\mathbf{b}})}{\sum_{b \in N} \mathrm{sim}(a,b)}$$

Item-Based Nearest Neighbor

Uses the similarity between items (and not users) to make predictions

Example:

- Look for items that are like Item5
- Take Alice ratings for these items to predict the rating for Item5

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

The Cosine and Adjusted Cosine Similarity Measures

Cosine Similarity

- Produces better results in item-to-item filtering
- Ratings are seen as vectors in n-dimensional space
- Similarity is calculated based on the angle between the vectors

$$sim(\vec{a},\vec{b}) = \frac{\vec{a}\cdot\vec{b}}{|\vec{a}| * |\vec{b}|}$$

Adjusted Cosine Similarity

- Takes average user ratings into account
- Transforms the original ratings

$$sim(\vec{a},\vec{b}) = \frac{\sum_{u\in U}(r_{u,a}-\overline{r_u})(r_{u,b}-\overline{r_u})}{\sqrt{\sum_{u\in U}(r_{u,a}-\overline{r_u})^2}\sqrt{\sum_{u\in U}(r_{u,b}-\overline{r_u})^2}}$$

n.vimalkumar@hotmail.com

Recommender Systems

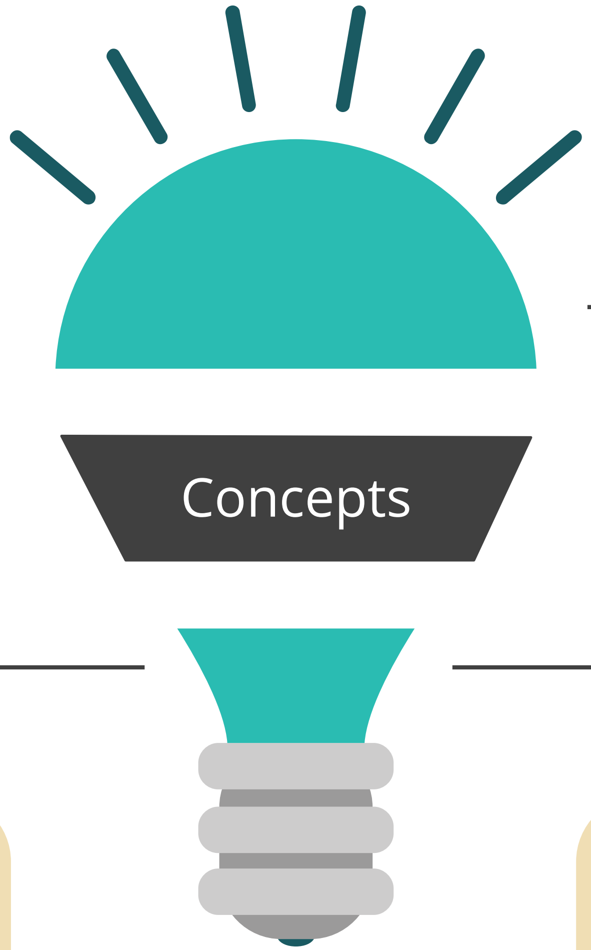
Topic 3: Association Rule Mining

n.vimalkumar@hotmail.com

Association Rule: Basic Concepts

Given a database of transactions, each transaction is considered as a list of items

Finds all rules that correlate the presence of one set of items with that of another set of items



Identifies frequent patterns

Most commonly used for market basket analysis

n.vimalkumar@hotmail.com

Association Rule: Performance Measures

Performance Measures

Support

- Provides fraction of transactions which contains the item X and Y
- Support = No. of times item X occurred / Total number of transactions = P (X ∪ Y))= $\frac{\sigma(x \cup y)}{N}$

Confidence

- Indicates how often the items X and Y occur together, given the no. of times X occurs
- Confidence = No. of times item X and Y occurred / Total occurrence of X = P(Y | X) = $\frac{\sigma(x \cup y)}{\sigma(x)}$

Lift

- Indicates the strength of a rule over the random co-occurrence of X and Y
- Lift = No. of times item X and Y occurred / Total occurrence of X multiplied by Total occurrence of Y = $\frac{\sigma(x \cup y)}{\sigma(x) \times \sigma(y)}$

n.vimalkumar@hotmail.com

Association Rule: Example

Suppose there are five transactions P1,P2,P3,P4,P5 as given below:

- P1 : A, B, C
- P2 : A, C, D
- P3 : B, C, D
- P4 : A, D, E
- P5 : B, C, E

Here,

- A,B,C,D,E are items in a store, I = {A,B,C,D,E}
- Set of all transactions P = {P1,P2,P3,P4,P5}
- Each transaction is a set of items, P ⊆ I

Association Rule: Example (Contd.)

Consider, you made some association rules using the transaction database as given below:

$A \Rightarrow D$
$C \Rightarrow A$
$A \Rightarrow C$
B and $C \Rightarrow D$

Calculating support, confidence, and lift values for the same will result in the following matrix:

Rule	Support	Confidence	Lift
$A \Rightarrow D$	2/5	2/3	2/9
$C \Rightarrow A$	2/5	2/4	1/6
$A \Rightarrow C$	2/5	2/3	1/6
B and $C \Rightarrow D$	1/5	1/3	1/9

Association Rule Generation: Apriori Algorithm

01 Uses frequent itemset to generate association rules

02 A subset of a frequent itemset must also be a frequent itemset

03 Frequent itemset is a set of items whose support value > threshold value

n.vimalkumar@hotmail.com

Apriori Algorithm: Example

Consider the following transaction dataset

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5
500	1 3 5

Minimum Support Count = 2

Threshold Value = 2

Apriori Algorithm: Step 01

List of one side itemsets are made and their support values are calculated:

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5
500	1 3 5

CI1

Itemset	Support
{1}	3
{2}	3
{3}	4
{4}	1
{5}	4

FI1

Itemset	Support
{1}	3
{2}	3
{3}	4
{5}	4

Since, the threshold value is 2, any itemset with support less than 2 are omitted.

n.vimalkumar@hotmail.com

Apriori Algorithm: Step 02

The length of the itemset is extended with 1 (k = k+1)

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5
500	1 3 5

CI2

Itemset	Support
{1,2}	1
{1,3}	3
{1,5}	2
{2,3}	2
{2,5}	3
{3,5}	3

FI2

Itemset	Support
{1,3}	3
{1,5}	2
{2,3}	2
{2,5}	3
{3,5}	3

n.vimalkumar@hotmail.com

Apriori Algorithm: Step 03

The length of the itemset is extended. All the combinations of itemsets in FI2 are used.

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5
500	1 3 5

CI3

Itemset	Support
{1,2,3}	
{1,2,5}	
{1,3,5}	
{2,3,5}	

Apriori Algorithm: Step 04

Divide your itemset to check if there are any other subsets whose support you haven't calculated yet.

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5
500	1 3 5

CI3

Itemset	In FI2?
{1,2,3} {1,2},{1,3},{2,3}	No
{1,2,5} {1,2},{1,5},{2,5}	No
{1,3,5} {1,5},{1,3},{3,5}	Yes
{2,3,5} {2,3},{2,5},{3,5}	Yes

FI2

Itemset	Support
{1,3}	3
{1,5}	2
{2,3}	2
{2,5}	3
{3,5}	3

n.vimalkumar@hotmail.com

Apriori Algorithm: Step 05

Using the itemsets of CI3, a new itemset CI4 is created.

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5
500	1 3 5

FI3

Itemset	Support
{1,3,5}	2
{2,3,5}	2

CI4

Itemset	Support
{1,2,3,5}	1

Since, support of CI4 is less than 2, you will stop and return to the previous itemset, CI3.

n.vimalkumar@hotmail.com

Apriori Algorithm: Step 06

Now, you will proceed with subset creation with the obtained list of frequent itemsets:

Itemset	Support
{1,3,5}	2
{2,3,5}	2

Confidence Value – 60%

Using this, you will generate all non-empty subsets for each frequent itemset:

- For I = {1,3,5}, subsets are {1,3}, {1,5}, {3,5}, {1}, {3}, {5}
- For I = {2,3,5}, subsets are {2,3}, {2,5}, {3,5}, {2}, {3}, {5}

For every subset S of I, output is:

- **S → (I-S)** (S recommends I-S)
- if support(I) / support(S) >= min_conf value

Apriori Algorithm: Rule Selection

Based on the threshold value, few rules are selected

For set {1,3,5}:

- Rule 1: {1,3} → ({1,3,5} - {1,3}) means 1 and 3 → 5

Confidence = support(1,3,5)/support(1,3) = 2/3 = 66.66% > 60%

Rule 1 is selected

- Rule 2: {1,5} → ({1,3,5} - {1,5}) means 1 and 5 → 3

Confidence = support(1,3,5)/support(1,5) = 2/2 = 100% > 60%

Rule 2 is selected

- Rule 3: {3,5} → ({1,3,5} - {3,5}) means 3 and 5 → 1

Confidence = support(1,3,5)/support(3,5) = 2/3 = 66.66% > 60%

Rule 3 is selected

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5
500	1 3 5

Apriori Algorithm: Rule Selection (Contd.)

For set {1,3,5}:

- Rule 4: {1} → ({1,3,5} - {1}) means 1 → 3 and 5

Confidence = support(1,3,5)/support(1) = 2/3 = 66.66% > 60%

Rule 4 is selected

- Rule 5: {3} → ({1,3,5} - {3}) means 3 → 1 and 5

Confidence = support(1,3,5)/support(3) = 2/4 = 50% <60%

Rule 5 is rejected

- Rule 6: {5} → ({1,3,5} - {5}) means 5 → 1 and 3

Confidence = support(1,3,5)/support(5) = 2/4 = 50% < 60%

Rule 6 is rejected

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5
500	1 3 5

Assisted Practice

Collaborative Filtering

Duration: 20 mins.

Problem Statement: Consider the ratings dataset below, containing the data on: UserID, MovieID, Rating and Timestamp. Each line of this file represents one rating of one movie by one user, and has the following format: UserID::MovieID::Rating::Timestamp
Ratings are made on a 5 star scale with half star increments.
UserID: represents ID of the user
MovieID: represents ID of the movie
Timestamp: represents seconds from midnight Coordinated Universal Time (UTC) of January 1, 1970.

Objective: Predict a User-movie recommendation model.

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

Unassisted Practice

Collaborative Filtering Duration: 15 mins.

Problem Statement: Consider the ratings dataset below, containing the data on: UserID, MovieID, Rating and Timestamp. Each line of this file represents one rating of one movie by one user, and has the following format:
UserID::MovieID::Rating::Timestamp
Ratings are made on a 5 star scale with half star increments.
UserID: represents ID of the user
MovieID: represents ID of the movie
Timestamp: represents seconds from midnight Coordinated Universal Time (UTC) of January 1, 1970.

Objective: Predict a movie-movie recommendation model.

Note: This practice is not graded. It is only intended for you to apply the knowledge you have gained to solve real-world problems.

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

Step 01

Load the ‘Ratings’ movie dataset into pandas with labels

Code

```
df = pd.read_csv('Recommend.csv',names=['user_id', 'movie_id', 'rating',
'timestamp'])
df
```

	user_id	movie_id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596
5	298	474	4	884182806
6	115	265	2	881171488
7	253	465	5	891628467
8	305	451	3	886324817

Step 02

Create a train test split of 75/25 by declaring number of users and movies

Code

```
n_users = df.user_id.unique().shape[0]
n_movies = df.movie_id.unique().shape[0]
train_data, test_data = train_test_split(df, test_size=0.25)
```

Step 03

Populate the train matrix (user_id x movie_id) with ratings such that [user_id index, movie_id index] = given rating

Code

```
train_data_matrix = np.zeros((n_users, n_movies))
for line in train_data.itertuples():
    #[user_id index, movie_id index] = given rating.
    train_data_matrix[line[1]-1, line[2]-1] = line[3]
train_data_matrix
```

```
array([[ 5.,  3.,  4., ...,  0.,  0.,  0.],
       [ 4.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       ...,
       [ 5.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  5.,  0., ...,  0.,  0.,  0.]])
```

Step 04

Populate the test matrix (user_id x movie_id) with ratings such that
[user_id index, movie_id index] = given rating

Code

```
test_data_matrix = np.zeros((n_users, n_movies))
for line in test_data.itertuples():
    #[user_id index, movie_id index] = given rating.
    test_data_matrix[line[1]-1, line[2]-1] = line[3]
test_data_matrix
```

```
array([[ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       ...,
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.]])
```

Step 05

Create cosine similarity matrices for movies and predict a movie-movie recommendation model

Code

```
movie_similarity = pairwise_distances(train_data_matrix.T,
metric='cosine’)
movie_pred = train_data_matrix.dot(movie_similarity) /
np.array([np.abs(movie_similarity).sum(axis=1)])
movie_pred
```

```
array([[ 0.37095251,  0.38716765,  0.40025082, ...,  0.44854253,
         0.4422732 ,  0.43775846],
       [ 0.09440623,  0.11043915,  0.10521177, ...,  0.11243308,
         0.11192467,  0.11294353],
       [ 0.06272422,  0.06545004,  0.06302051, ...,  0.06365259,
         0.06343483,  0.06429965],
       ...,
       [ 0.02838429,  0.03586421,  0.03477146, ...,  0.04045211,
         0.03992107,  0.03988944],
       [ 0.12626427,  0.13544609,  0.14149739, ...,  0.14753123,
         0.14742251,  0.14737875],
       [ 0.2047876 ,  0.19941526,  0.2216142 , ...,  0.25104105,
```

Key Takeaways

Now, you are able to:

- Build recommender model using python
- Understand mechanism of association rule mining
- Demonstrate apriori algorithm


?
Knowledge
Check

Knowledge Check 1	**Which of the following would most indicate a situation, where user-user collaborative filtering would be strongly preferable for content-content based filtering?**

a. The items recommended don’t have good attributes or keywords to describe them.

b. Only implicit ratings are available.

c. Most users have rated a core set of popular items, though they have different tastes on that set.

d. There are lots of items to recommend and relatively few users.

n.vimalkumar@hotmail.com

Knowledge Check 1	Which of the following would most indicate a situation, where user-user collaborative filtering would be strongly preferable for content-content based filtering?

a. **The items recommended don't have good attributes or keywords to describe them.**

b. **Only implicit ratings are available.**

c. **Most users have rated a core set of popular items, though they have different tastes on that set.**

d. There are lots of items to recommend and relatively few users.

The correct answer is **a. The items recommended don't have good attributes or keywords to describe them.**

User-user collaborative filtering is strongly preferable when, the items recommended don't have good attributes or keywords to describe them or less number of items.

Knowledge Check

2

Which of the following is not a requirement for a successful user-user collaborative filtering system?

a. Users tastes must either be stable (individual) or changing. If changing, they change in sync with other users' tastes.

b. The domain in which you are performing collaborative filtering is scoped such that people who agree within one part of that domain generally agree within other parts of the domain.

c. Past agreement between users is predictive of future agreement.

d. Users mostly have similar tastes on a set of popular items, though they may have individually different tastes on unpopular items.

Knowledge Check

2

Which of the following is not a requirement for a successful user-user collaborative filtering system?

a. Users tastes must either be stable (individual) or changing. If changing, they change in sync with other users' tastes.

b. The domain in which you are performing collaborative filtering is scoped such that people who agree within one part of that domain generally agree within other parts of the domain.

c. Past agreement between users is predictive of future agreement.

d. Users mostly have similar tastes on a set of popular items, though they may have individually different tastes on unpopular items.

The correct answer is **d. Users mostly have similar tastes on a set of popular items, though they may have individually different tastes on unpopular items.**

If the users mostly have similar tastes on a set of popular items, it calls for hybrid or item-based collaborative filtering.

Lesson-End Project

Duration: 20 mins.

Problem Statement: BookRent is the largest online and offline book rental chain in India. The company charges a fixed and rental fee for a book per month. Hence, the company gets more money for rented books. Since, most of the users returning the books and not taking rentals, the company wants to increase the revenue and profit.

Objective: You as an ML expert have to model a recommendation engine so that user gets recommendations of books based on the behaviour of similar users. This will ensure that users are renting books based on their individual taste.

Access: Click the Labs tab in the left side panel of the LMS. Copy or note the username and password that are generated. Click the Launch Lab button. On the page that appears, enter the username and password in the respective fields and click Login.

Thank You

n.vimalkumar@hotmail.com